Details of Construction of Games Grid Board

This invention is related to electronic grid games boards, in which players play by pressing a point in a grid of points on an electronic board, and the board displays the state of the game by illuminating some of the points. It describes a construction to achieve an implementation of the grid itself which is cheap and robust, feel good to play on and is visually attractive.

Electronic games boards like the ones which are described by Harpaz(1999, GB 9919551.3) are played by the users pressing points on the board, and in respond the board changes the illumination of some points. For their implementation, they require a grid of points (each of which will be referred to as *a grid point* in this text), each of which can sense when it is pressed, and can be illuminated by different colours. In addition, it needs some controller that *manages* the games and puzzles that the board can play, which means it is notified when a point is pressed, and changes the illumination of the points according to the rules of the current game. This invention is concerned with the implementation of the grid points.

The implementation of the grid points needs to be robust and cheap, feel good to play on and be visually appealing. In general, previous publications mostly assume that the illuminations will be done by LEDs, and the sensing of a press will be done either by a touch screen or by mechanical buttons. See for example:

Zeki *Et al*	International Publication Number	WO 97/26057
Weiner *et al*	US patent	5,573,245
Golad	International Publication Number	WO 98/19758

A touch screen seems to be too expansive and damage-sensitive to be really useful, and the few small toys which are already on the market and have such points seem to be all made of buttons. These are also relatively expensive, suffer mechanical wear, and it is difficult to illuminate the buttons in an attractive way. Additional problem with buttons is that they have to be pressed vertically, and when playing fast that is a problem for the players, because they have to move their hand horizontally to the right place and stop it to actually press the point.

A possible solution with current technology is to use membrane keyboard for the input, which solves the main problems of robustness and is relatively cheap. However, simply using a membrane keyboard above point light sources like LEDs does not look appealing. The main problems are that the illuminated area is not uniformly illuminated (it is brighter just above the light source, and there are also smaller inhomogeneities), and that if more than one light source is used, they illuminate somewhat different areas. The invention here describes a construction that gives an almost uniformly illuminated area, with more than one light source, and is based on standard and cheap components.

According to the current inventions, the grid of points is constructed from the following four layers:

1) A membrane keyboard with a regular grid of press-sensitive areas, which is opaque except for a transparent circular area for each of the press-sensitive areas.

2) A layer of a uniform, translucent material below the membrane keyboard.

3) A separation layer of opaque material below the translucent layer, which for each grid point has a transparent (typically empty) circular hole matching the transparent circular areas in the membrane keyboard, with reflecting or very light walls.

4) A PCB below the separation layer, on which two or more light sources per grid point are mounted, and also carries the circuitry to drive them.

The four layers are held together, and the thickness of the separation layer is such that the light sources are not immediately below the translucent area, but some distance away. The distance needs to be more than quarter of the diameter of the hole in the separation layer, and would typically be 40-140% of the diameter of the hole. The membrane keyboard and the control of the light sources are both connected to a CPU with some memory which uses them to manage various games and puzzles.

Because the sources of light are not immediately below the translucent layer, the light spreads before it reaches the translucent layer, so the difference between the illumination in the centre of the illuminated area and in its periphery is not so marked. The sides of the holes reflect light that doesn't reach the translucent layer directly, and this reflected light is distributed homogeneously or even more at the periphery of the illuminated area, so makes the distribution of illumination more homogeneous. The translucent layer smoothes out short range changes in the illumination.

The overall result is illumination that still decrease from the point above the source of light, but the change is small and smooth. Outside the circular transparent areas in the graphics layer of the membrane keyboard illumination drops to zero, thus giving a circular and almost homogeneous illuminated area.

The membrane keyboard will also need some marking which show the players where the grid points are even when they are not illuminated.

In contrast to keyboards that are used in working devices, for a playing board it is important that it is very easy to press a point, even if this leads to more erroneous presses. This allows the players to play fast and without concentrating on the accuracy of the movements that they perform. A possible way to achieve this is by making the press-sensitive area for each grid point much larger than the illuminated area.

Another possibility is to make the membrane keyboard with a dome for each point, rising above the surface. That makes it easier for the players to feel where they press. With the current technology it seems difficult to combine such domes with a large pressing area, but advances in the membrane keyboard technology may make it plausible in the near future.

The grid that is shown in the specific embodiment is a square grid, but the grid can have other arrangement, e.g. hexagonal, triangular or rectangular.

In the separation layer, the important part for the invention are just the cylinders around each point. The space between the cylinders maybe completely empty, completely full, or any other possibility between these extremes.

A specific embodiment of the invention will now be described with reference to the accompanying drawings:

Figure 1 Shows a sketch of a typical board from above
Figure 2 Shows a cross section through the a column of grid points in the board
Figure 3 Shows an expansion of one grid point.

Figure 1 shows the grid from above, so only the membrane keyboard 1 and the illumination of grid points 3 are visible. Unilluminated points are denoted by empty circles, and illuminated points are denoted one of two patterns, one for red and one for green. The membrane keyboard contains an obliterating gray layer 2, with a circular hole for each grid point. The translucent layer 4 is made of HIPS (High Impact PolyStyrene) . The layout of the membrane keyboard is specially designed for the board, but otherwise it is made by standard techniques.

The separation layer 5 is made of an injected material. The walls of the holes 6 are smooth, so they also reflect some of the light. The space between the cylinders 12 is empty, except a connecting layer just below the translucent layer.

The PCB 7 is a standard PCB, and for each grid point it has two LEDs 8, one green and one red, and are mounted to project their light upwards.

The actual shape of the all the layers is a rectangle, such that they extend more than the square grid of points to one direction. The CPU and memory 9 are mounted above the PCB at this region, and on the top there are input and display devices 10 to allow the players to control the games that the CPU manages. This region also contains the electronic connection between the PCB and the membrane keyboard and the power input (not shown).

The LEDs produce a cone of light 11, and the thickness of the separation layer is such that the cone of light intersects the walls of the hole a small distance below the translucent layer. With this construction, the periphery of the illuminated area receives more reflected light from the top of the walls than the centre, thus increasing the homogeneity of the illumination. The LEDs are tilted slightly towards the centre, to compensate for the fact that they are not exactly in the middle of the hole. The two LEDs in each grid point can be

switched on independently, so the grid points can be illuminated either by green or red , which are marked with different patterns in Fig 1.

The membrane keyboard is glued to the translucent layer, which is glued to the separation layer. The separation layer is connected by screws to another layer 13 of the same material, and together they hold the PCB 7 and form a box around it.

The press-sensitive area for one grid point is indicated in Figure 1 by the square in broken line 14. As discussed above it is much larger than the illuminated area of the grid point, to make it easier for the players to press a point.

The visual look of the board and the easiness of making moves make it very appealing for playing any game that can be played on it. A list of such games is given below. It is even better for games when time is important, either because the players have a limited time to make a move, or because the illumination of the points changes even when the players don't make a move, like in the games *Ghost, TouchIt, Life1* and *Life2* below(Life is a RTM). With existing constructions, these games would not be enjoyable to play.

Games implemented in the prototype of the Grid Board

The following 12 games have been implemented in the prototype of the board. Each game has several parameters that can be set, but only a minority of these are described here. Each two-players game can also be played against the board, and has a time limit for performing a move, which can be set by the players. With the default parameters settings, *Othello, CountLines* and *Visiput* flash the legal moves for a player that presses a point that is an illegal move.

1) *Othello.* 'Adding a stone' is done by pressing an unilluminated point. If this is a legal move, the CPU switches the point on with the current player's colour (which corresponds to putting a stone in this point) and reverses the colour of the points that need to be reversed according to the rules of OTHELLO. Because the number of points is 9x9 rather than the usual 8x8, the start up position is different from the standard starting position. In each move the CPU checks if the current player has a legal move, and if not passes the turn to the other player. If both players don't have a legal move, the CPU finishes the game. In OTHELLO, passing a move is actually is illegal, but the time limit of a move means that a player may intentionally pass, which may be regarded as cheating. To get over this problem, a parameter called *Compensate* can be set, so when a player passes a move, the other player gets as a compensation more than one move.

2) *Ghost.* This is a fluid game. The board illuminates four points (the *Ghost*), and then moves the *Ghost*, by repeatedly switching on a point that is a neighbour of one of the illuminated points, and switching off one of the illuminated points. The players try to 'catch the *Ghost*', by pressing one of the points while it is illuminated

3) *Life1.*This is a fluid game, i.e. the state of the illumination of grid points changes even if the players do not play. Each fixed time period (*generation*, a settable parameter) , the CPU

checks for each point how many of the eight points around it are illuminated, and accordingly decides if the point is going to be illuminated in the next generation. Thus the pattern of illumination of the grid points changes each generation. In parallel, the player(s) can switch on or switch off points by pressing them.

Life1 can be played in a 'kill' mode, in which the player tries to 'kill' the board, i.e. switch off all the points, as fast as possible, or in 'keep alive' mode, in which the player tries to keep the board 'alive', i.e. keep at least some points on, as long as possible. Adjusting the various parameters makes the task an interesting challenge.

4) *Life2*. Like *Life1*, but there are points illuminated in either colour. The players play in turn, and each player tries to switch off all the points of the other colour.

5) *ToucIt*. Single player game, mainly testing reaction time and accuracy. The CPU switches on a few points (1-4) one after the other with a short time gap, and then switches them off. The player needs to press the last point that was switched on before the next point is switched on or all of them are switched off. By changing the time gap between switching the points on, the number of points and their pattern, the player can fit the game to his own level to make it a good challenge.

6) *Symmetry*. Single player game. The CPU switches on a pattern of points on one side of the grid, and the player needs to press the symmetry related points on the other side of the grid. Parameters like the number of points in the pattern, the time that is allowed for doing the copying and the kind of symmetry operation that the player need to do are used to match the difficulty level of the copying to the level of the player.

7) *ClearIt*. The game starts with an equal number of points illuminated in each of the two colours, but otherwise randomly distributed on the grid. Each player in his turn presses an empty point, and the CPU switches of all the points of the other player's colour which are a chess knight move away from the pressed point, and points of the current player's colour which are one diagonal move further. The winner is the player that switches off all the other player's points first.

8) *FindThem*. A memory game. In the beginning of the game the CPU switches on an equal number of points in both colours but in a random pattern for a short period of time, and then switches all of them off. Each player in turn tries to switch on a point of his own colour by pressing a point. If this point was switched on with his colour in the beginning, it is switched on. The winner is the player that first switches on 9 points.

9) *CountLines*. Each player in his turn switches on a point of his colour by pressing it. A player is not allowed to press in two successive turns two points that are too close to each other, and the distance that defines what is too close can be set by the players. The winner is the player that when all the points are switched on has the larger number of straight lines of four points of his colour.

10) *Visiput*. Each player in his turn switches on a point by pressing it. A player is allowed to press a point only if the point is 'visible' by points of his own colour at least as it is 'visible' by points of the other colour, where 'visible' means connected by a straight line of switched-off points. The winner is the player that succeeds to switch on more points. The CPU checks the games after each move, and if it can determine for each unilluminated point that only one player would be able to press it, it switches on all the unilluminated points to the colour of the player that can press it, and finishes the game, thus saving the players the boring task of switching all the unilluminated points after the result of the game is already known.

11) *Go*. The game is played according to the standard rules of GO. For each move, the board checks if it is legal, and rejects it otherwise. If the move captures points of the other players, the board switches them off. Like in *Visiput*, the CPU finishes the game when it is clear which part of the grid is controlled by whom. Because this is a complex operation in Go and the heuristics that are used may get it wrong, this feature is controlled by several settable parameters, and can also be switched off.

12) *FillIt*. Each player in his turn presses an empty point which causes a pattern of points around this point to be switched on with the player colour if they were off, or reverse their colour if they were on. The winner is the player that has more points when all the points are switched on.